Introduction to Hidden Markov Model and Its Application

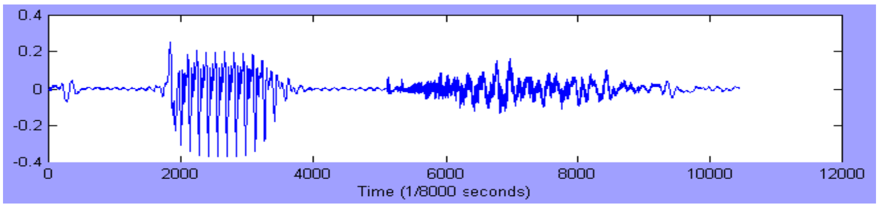
April 16, 2005

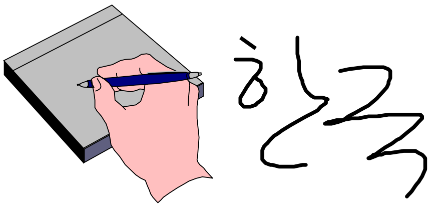
Dr. Sung-Jung Cho
sung-jung.cho@samsung.com
Samsung Advanced Institute of Technology (SAIT)

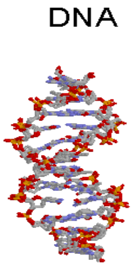
Contents

Sequential Data

- Data are sequentially generated according to time or index
- Spatial information along time or index







```
AAAAGAAAAGGTTAGAAAGATGAGAGATGATAAAGGGTCCATTTG
AGGTTAGGTAATATGGTTTGGTATCCCTGTAGTTAAAAGTTTTTG
TCTTATTTTAGAATACTGTGACTATTTCTTTAGTATTAATTTTTC
CTTCTGTTTTCCTCATCTAGGGAACCCCAAGAGCATCCAATAGAA
GCTGTGCAATTATGTAAAATTTTCAACTGTCTTCCTCAAAATAAA
GAAGTATGGTAATCTTTACCTGTATACAGTGCAGAGCCTTCTCAG
AAGCACAGAATATTTTTATATTTCCTTTATGTGAATTTTTAAGCT
GCAAATCTGATGGCCTTAATTTCCTTTTTGACACTGAAAGTTTTG
TAAAAGAAATCATGTCCATACACTTTGTTGCAAGATGTGAATTAT
TGACACTGAACTTAATAACTGTGTACTGTTCGGAAGGGGTTCCTC
AAATTTTTTGACTTTTTTTGTATGTGTGTTTTTTCTTTTTTTTTA
AGTTCTTATGAGGAGGGAGGGTAAATAAACCACTGTGCGTCTTGG
TGTAATTTGAAGATTGCCCCATCTAGACTAGCAATCTCTTCATTA
TTCTCTGCTATATATATAAAACGGTGCTGTGAGGGAGGGGAAAAGCA
TTTTTCAATATATTGAACTTTTGTACTGAATTTTTTTGTAATAAG
CAATCAAGGTTATAATTTTTTTTAAAATAGAAATTTTGTAAGAAG
GCAATATTAACCTAATCACCATGTAAGCACTCTGGATGATGGATT
CCACAAAACTTGGTTTTATGGTTACTTCTTCTCTTAGATTCTTAA
TTCATGAGGAGGGTGGGGGAGGGAGGTGGAGGGAGGGAAGGGTTT
CTCTATTAAAATGCATTCGTTGTGTTTTTTAAGATAGTGTAACTT
GCTAAATTTCTTATGTGACATTAACAAATAAAAAAGCTCTTTTAA
TATTAGATAA
```

18

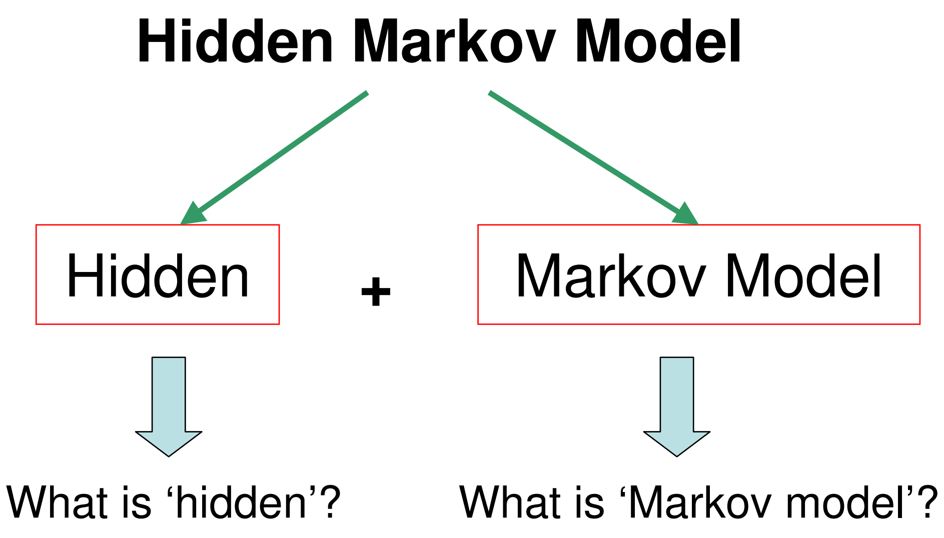
Advantage of HMM on Sequential Data

- Natural model structure: doubly stochastic process
  - transition parameters model *temporal* variability
  - output distribution model *spatial* variability
- Efficient and good modeling tool for
  - sequences with temporal constraints
  - spatial variability along the sequence
  - real world complex processes
- Efficient evaluation, decoding and training algorithms
  - Mathematically strong
  - Computaionally efficient
- Proven technology!
  - Successful stories in many applications

Successful Application Areas of HMM

- On-line handwriting recognition
- Speech recognition
- Gesture recognition
- Language modeling
- Motion video analysis and tracking
- Protein sequence/gene sequence alignment
- Stock price prediction
- …

What's HMM?



Hidden Markov Model

Hidden + Markov Model

What is 'hidden'?

What is 'Markov model'?

Markov Model

- Scenario
- Graphical representation
- Definition
- Sequence probability
- State probability

Markov Model: Scenario

- Classify a weather into three states
  - State 1: rain or snow
  - State 2: cloudy
  - State 3: sunny

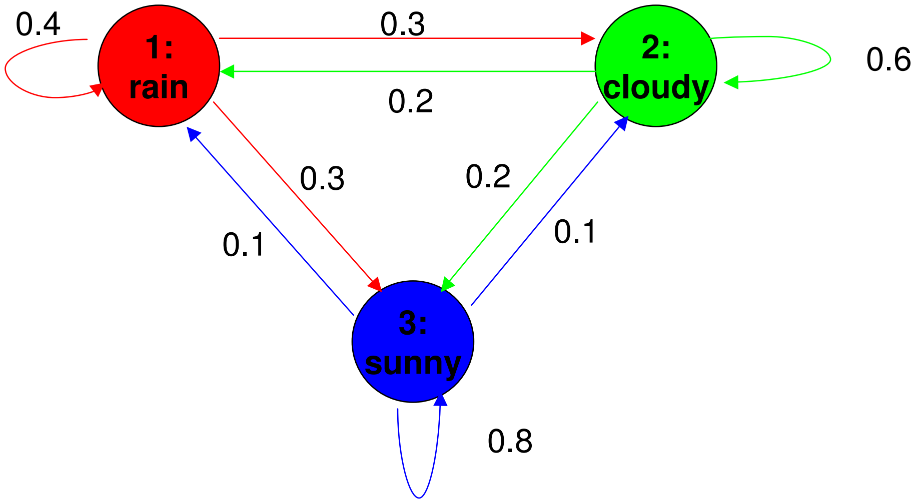
- By carefully examining the weather of some city for a long time, we found following weather change pattern

		Tomorrow		
		Rain/snow	Cloudy	Sunny
Today	Rain/Snow	0.4	0.3	0.3
	Cloudy	0.2	0.6	0.2
	Sunny	0.1	0.1	0.8

Assumption: tomorrow weather depends only on today one!

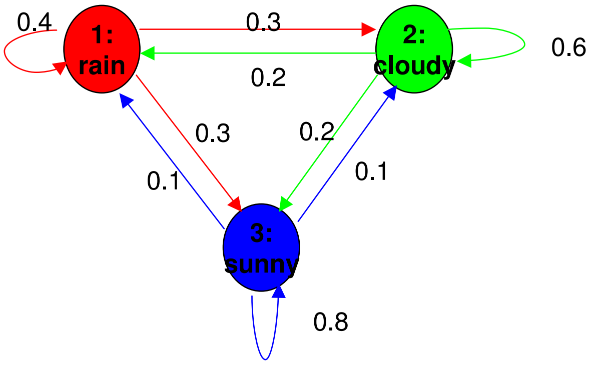
Markov Model: Graphical Representation

- Visual illustration with diagram



- Each state corresponds to one observation
- Sum of outgoing edge weights is one

Markov Model: Definition



- Observable states

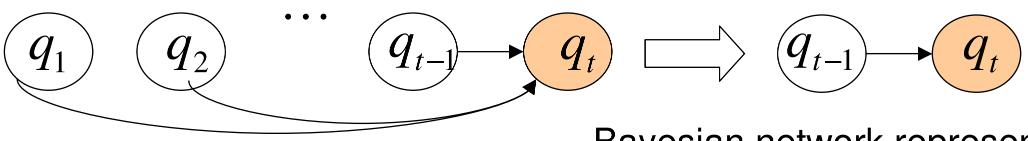
$$\{1,\ 2,\ \cdots,\ N\}$$

- Observed sequence

$$q_1, q_2, \cdots, q_T$$

- 1st order Markov assumption

$$P(q_t = j \mid q_{t-1} = i, q_{t-2} = k, \cdots) = P(q_t = j \mid q_{t-1} = i)$$

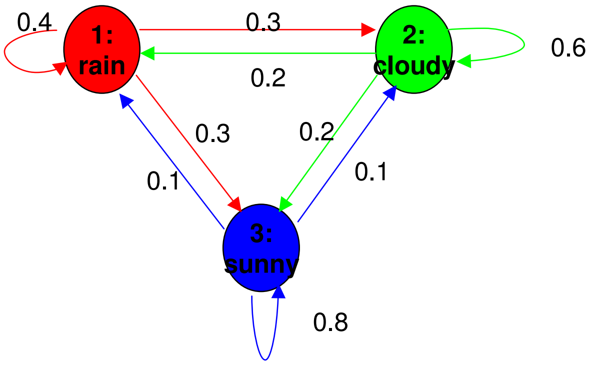


Bayesian network representation

- Stationary

$$P(q_t = j \mid q_{t-1} = i) = P(q_{t+l} = j \mid q_{t+l-1} = i)$$

Markov Model: Definition (Cont.)



- State transition matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ a_{N1} & a_{NN} & \cdots & a_{NN} \end{bmatrix}$$

  - Where

$$a_{ij} = P(q_t = j \mid q_{t-1} = i), \qquad 1 \le i, j \le N$$

  - With constraints

$$a_{ij} \ge 0, \qquad \sum_{j=1}^{N} a_{ij} = 1$$

- Initial state probability

$$\pi_i = P(q_1 = i), \qquad 1 \le i \le N$$

Markov Model: Sequence Prob.

- Conditional probability

$$P(A, B) = P(A \mid B) P(B)$$

- Sequence probability of Markov model

Chain rule

$$\begin{aligned} & P(q_1, q_2, \cdots, q_T) \\ &= P(q_1)P(q_2 \mid q_1) \cdots P(q_{T-1} \mid q_1, \cdots, q_{T-2}) P(q_T \mid q_1, \cdots, q_{T-1}) \\ &= P(q_1)P(q_2 \mid q_1) \cdots P(q_{T-1} \mid q_{T-2}) P(q_T \mid q_{T-1}) \end{aligned}$$

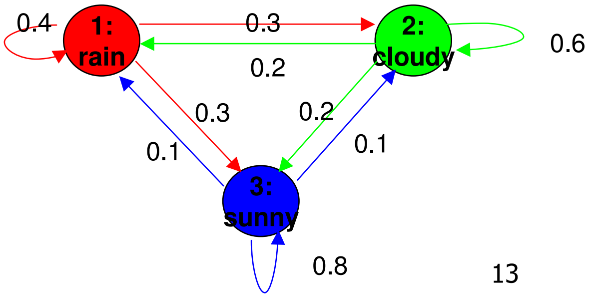
1st order Markov assumption

Markov Model: Sequence Prob. (Cont.)

- Question: What is the probability that the weather for the next 7 days will be "sun-sun-rain-rain-sun-cloudy-sun" when today is sunny?

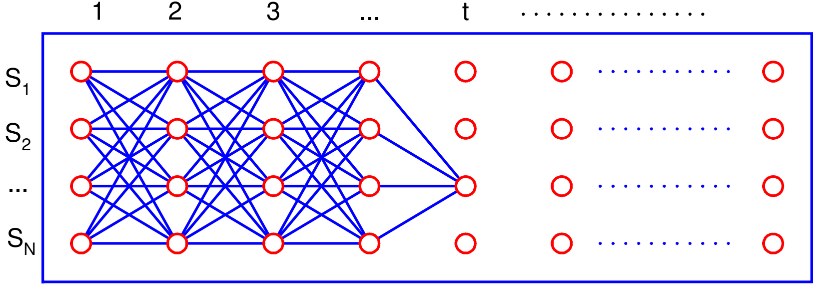
$$S_1 : rain, \quad S_2 : cloudy, \quad S_3 : sunny$$

$$\begin{aligned} P(O \mid \text{model}) &= P(S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3 \mid \text{model}) \\ &= P(S_3) \cdot P(S_3 \mid S_3) \cdot P(S_3 \mid S_3) \cdot P(S_1 \mid S_3) \\ &\quad \cdot P(S_1 \mid S_1) P(S_3 \mid S_1) P(S_2 \mid S_3) P(S_3 \mid S_2) \\ &= \pi_3 \cdot a_{33} \cdot a_{33} \cdot a_{31} \cdot a_{11} \cdot a_{13} \cdot a_{32} \cdot a_{23} \\ &= 1 \cdot (0.8)(0.8)(0.1)(0.4)(0.3)(0.1)(0.2) \\ &= 1.536 \times 10^{-4} \end{aligned}$$



Markov Model: State Probability

- State probability at time t : $P(q_t = i)$



- Simple but slow algorithm:
  - Probability of a path that ends to state *i* at time *t*:

$$Q_t(i) = (q_1, q_2, \cdots, q_t = i)$$

$$P(Q_t(i)) = \pi_{q_1} \prod_{k=2}^{t} P(q_k \mid q_{k-1})$$

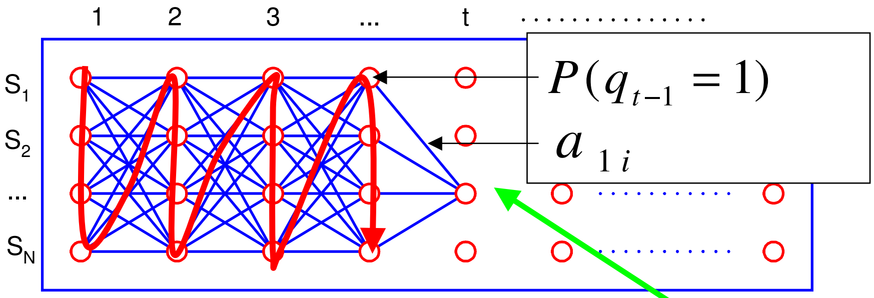
  - Summation of probabilities of all the paths that ends to I at t

$$P(q_t = i) = \sum_{all \ \ Q_t(i)'s} P(Q_t(i))$$

Exponential time complexity: $O(N^t)$

Markov Model: State Prob. (Cont.)

- State probability at time t : $P(q_t = i)$



Each node stores the sum of probabilities of partial paths

- Efficient algorithm
  - Recursive path probability calculation

$$P(q_t = i) = \sum_{j=1}^{N} P(q_{t-1} = j, q_t = i)$$

$$= \sum_{j=1}^{N} P(q_{t-1} = j) P(q_t = i \mid q_{t-1} = j)$$

$$= \sum_{j=1}^{N} P(q_{t-1} = j) \cdot a_{ji}$$

Time complexity: $O(N^2 t)$

What's HMM?

Hidden Markov Model

Hidden + Markov Model

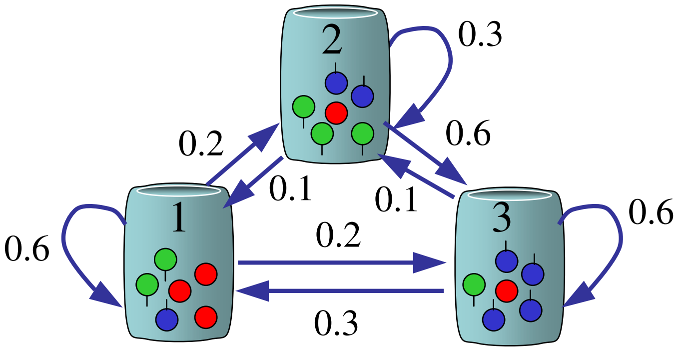
What is 'hidden'?

What is 'Markov model'?

Hidden Markov Model

- Example
- Generation process
- Definition
- Model evaluation algorithm
- Path decoding algorithm
- Training algorithm

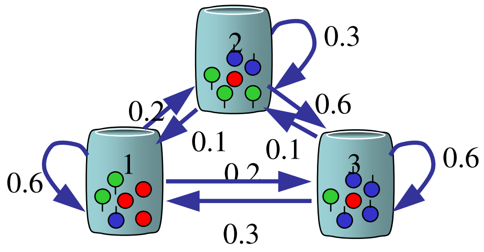
Hidden Markov Model: Example

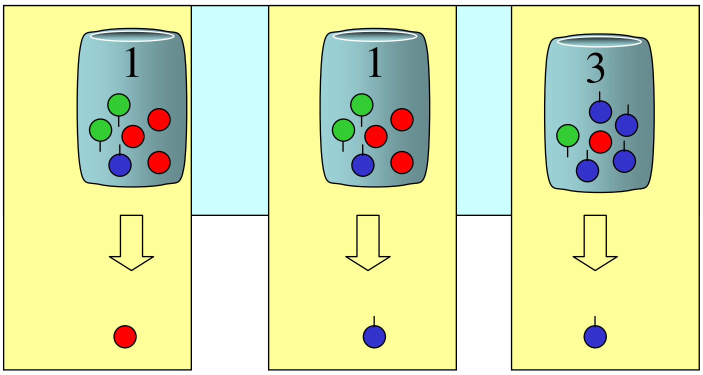


- N urns containing color balls
- M distinct colors
- Each urn contains different number of color balls

HMM: Generation Process

- Sequence generating algorithm
  - Step 1: Pick initial urn according to some random process
  - Step 2: Randomly pick a ball from the urn and then replace it
  - Step 3: Select another urn according to a random selection process
  - Step 4: Repeat steps 2 and 3



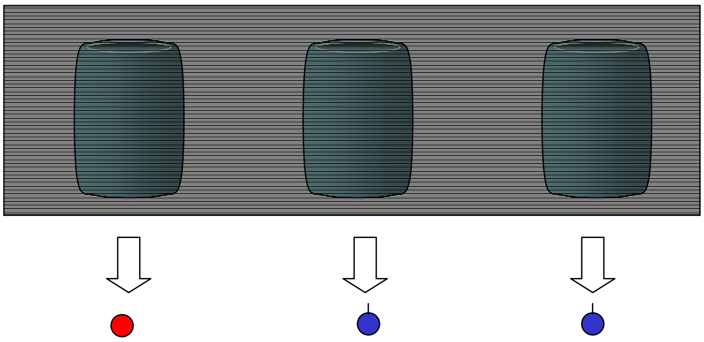


Markov process: $\{q(t)\}$

Output process: $\{f(x|q)\}$

HMM: Hidden Information

- Now, what is hidden?



  - We can just see the chosen balls
  - We can't see which urn is selected at a time
  - So, urn selection (state transition) information is hidden

HMM: Definition

- Notation: $\lambda = (\mathrm{A}, \mathrm{B}, \Pi)$
  - (1) N: Number of states
  - (2) M: Number of symbols observable in states

$$V = \{ v_1, \cdots, v_M \}$$

  - (3) A: State transition probability distribution

$$A = \{ a_{ij} \}, \quad 1 \le i, j \le N$$

  - (4) B: Observation symbol probability distribution

$$B = \{ b_i(v_k) \}, \quad 1 \le i \le N, 1 \le j \le M$$

  - (5) Π: Initial state distribution

$$\pi_i = P(q_1 = i), \quad 1 \le i \le N$$

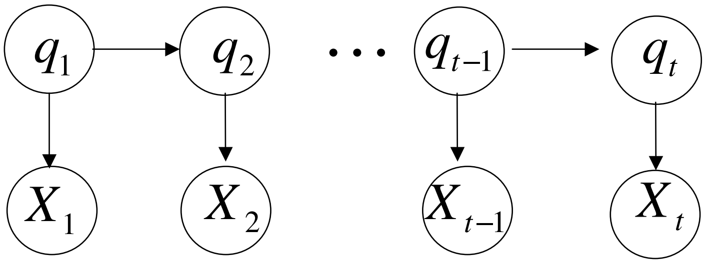
HMM: Dependency Structure

- 1-st order Markov assumption of transition

$$P(q_t \mid q_1, q_2, \cdots, q_{t-1}) = P(q_t \mid q_{t-1})$$

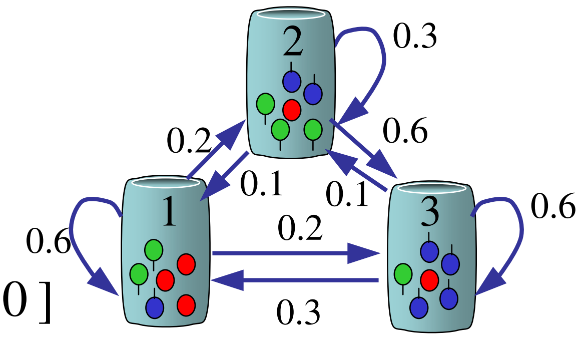
- Conditional independency of observation parameters

$$P(X_t \mid q_t, X_1, \cdots, X_{t-1}, q_1, \cdots, q_{t-1}) = P(X_t \mid q_t)$$



Bayesian network representation

HMM: Example Revisited



- # of states: N=3
- # of observation: M=3

  V = { R, G, B }
- Initial state distribution

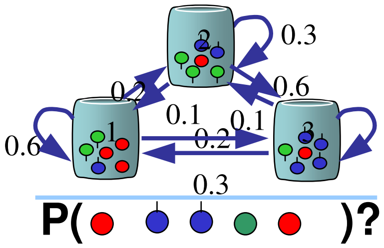
$$\pi = \{P(q_1 = i)\} = [1,0,0]$$

- State transition probability distribution

$$A = \{a_{ij}\} = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.1 & 0.3 & 0.6 \\ 0.3 & 0.1 & 0.6 \end{bmatrix}$$

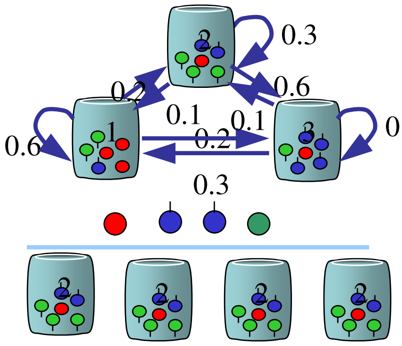
- Observation symbol probability distribution
- $$B = \{b_i(v_k)\} = \begin{bmatrix} 3/6 & 2/6 & 1/6 \\ 1/6 & 3/6 & 2/6 \\ 1/6 & 1/6 & 4/6 \end{bmatrix}$$

HMM: Three Problems



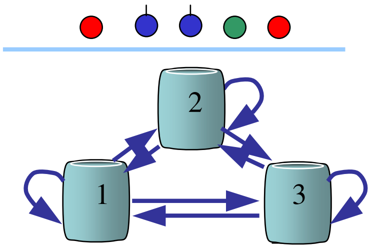
- What is the probability of generating an observation sequence?
  - Model evaluation

$$P(X = x_1, x_2, \cdots, x_T \mid \lambda) = ?$$



- Given observation, what is the most probable transition sequence?
  - Segmentation or path analysis

$$Q^* = \arg\max_{Q=(q_1,\cdots,q_T)} P(Q, X \mid \lambda)$$



- How do we estimate or optimize the parameters of an HMM?
  - Training problem

$$P(X \mid \lambda = (A, B, \pi)) < P(X \mid \lambda' = (A', B', \pi'))$$

Model Evaluation

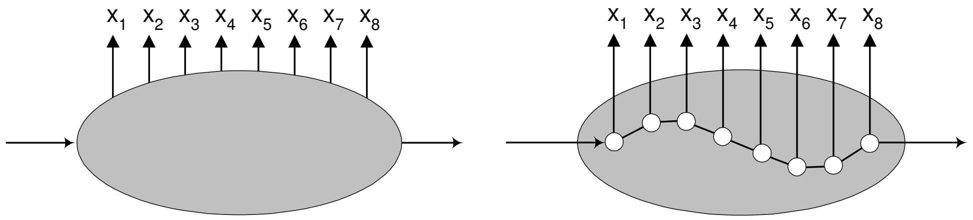
Forward algorithm
Backward algorithm

Definition

- Given a model λ
- Observation sequence: $X = x_1, x_2, \cdots, x_T$
- P(X| λ) = ?

- $P(X \mid \lambda) = \sum_{Q} P(X, Q \mid \lambda) = \sum_{Q} P(X \mid Q, \lambda) P(Q \mid \lambda)$

(A path or state sequence: $Q = q_1, \cdots, q_T$)



Solution

- Easy but slow solution: exhaustive enumeration

$$P(X \mid \lambda) = \sum_{Q} P(X, Q \mid \lambda) = \sum_{Q} P(X \mid Q, \lambda) P(Q \mid \lambda)$$

$$= \sum_{Q} b_{q_1}(x_1) b_{q_2}(x_2) \cdots b_{q_T}(x_T) \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \cdots a_{q_{T-1} q_T}$$

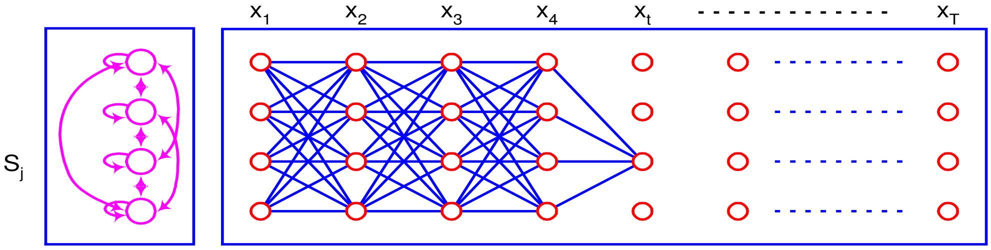
  - Exhaustive enumeration = combinational explosion!

$$O(N^T)$$

- Smart solution exists?
  - Yes!
  - Dynamic Programming technique
  - Lattice structure based computation
  - Highly efficient -- linear in frame length

Forward Algorithm

- Key idea
  - Span a lattice of N states and T times
  - Keep the sum of probabilities of all the paths coming to each state i at time t



- Forward probability

$$\alpha_t(j) = P(x_1 x_2 \ldots x_t, q_t = S_j \mid \lambda)$$

$$= \sum_{Q_t} P(x_1 x_2 \ldots x_t, Q_t = q_1 \ldots q_t \mid \lambda)$$

$$= \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(x_t)$$

Forward Algorithm

- Initialization

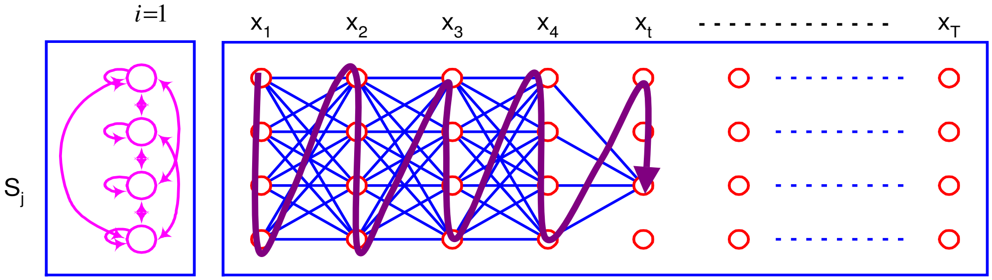
$$\alpha_1(i) = \pi_i b_i(\mathbf{x}_1) \qquad 1 \le i \le N$$

- Induction

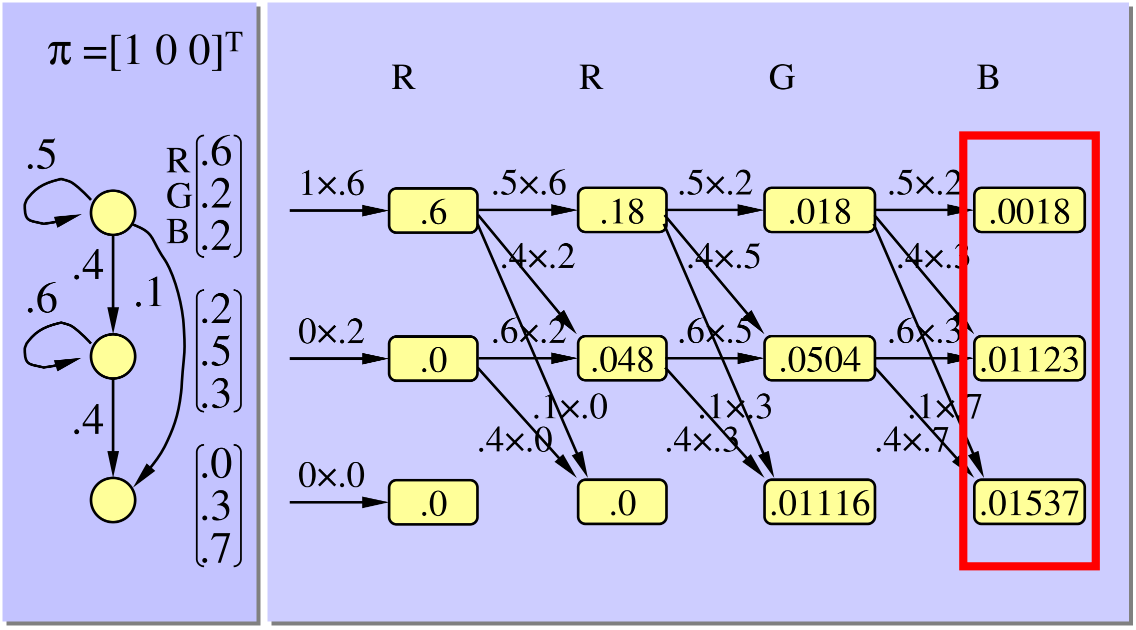
$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(\mathbf{x}_t) \qquad 1 \le j \le N,\ t = 2, 3, \cdots, T$$

- Termination

$$P(\mathbf{X} \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

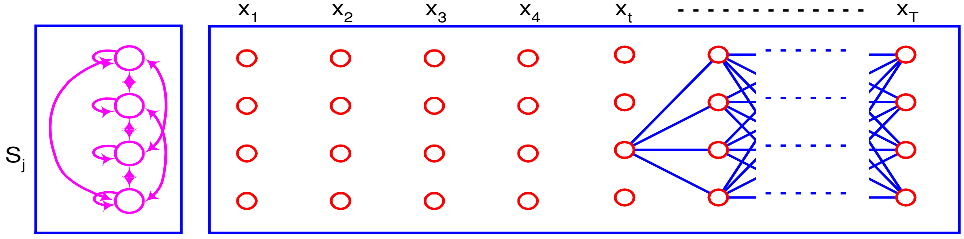


Numerical Example: $P(\text{RRGB}|\lambda)$ [신봉기 03]



Backward Algorithm (1)

- Key Idea
  - Span a lattice of N states and T times
  - Keep the sum of probabilities of all the outgoing paths at each state i at time t



- Backward probability

$$\beta_t(i) = P(x_{t+1}x_{t+2}\ldots x_T \mid q_t = S_i, \lambda)$$

$$= \sum_{Q_{t+1}} P(x_{t+1}x_{t+2}\ldots x_T, Q_{t+1} = q_{t+1}\ldots q_T \mid q_t = S_i, \lambda)$$

$$= \sum_{j=1}^{N} a_{ij} b_j(x_{t+1}) \beta_{t+1}(j)$$

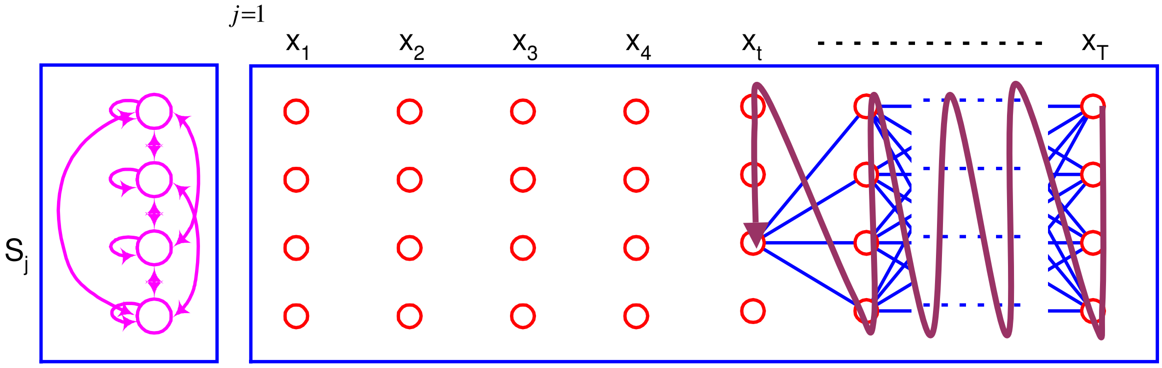
Backward Algorithm (2)

- Initialization

$$\beta_T(i) = 1 \qquad 1 \le i \le N$$

- Induction

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\mathbf{x}_{t+1}) \beta_{t+1}(j) \qquad 1 \le i \le N, \quad t = T-1, T-2, \cdots, 1$$

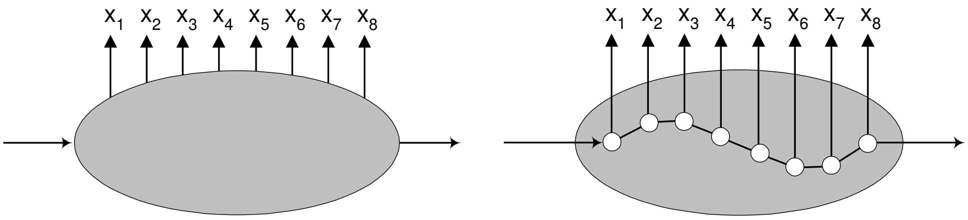


The Most Probable Path Decoding

State sequence
Optimal path
Viterbi algorithm
Sequence segmentation

The Most Probable Path

- Given a model λ
- Observation sequence: $X = \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T$
- $P(X, Q \mid \lambda)$ = ?
- $Q^* = \arg\max_Q P(X, Q \mid \lambda) = \arg\max_Q P(X \mid Q, \lambda) P(Q \mid \lambda)$
  - (A path or state sequence: $Q = q_1, \cdots, q_T$)



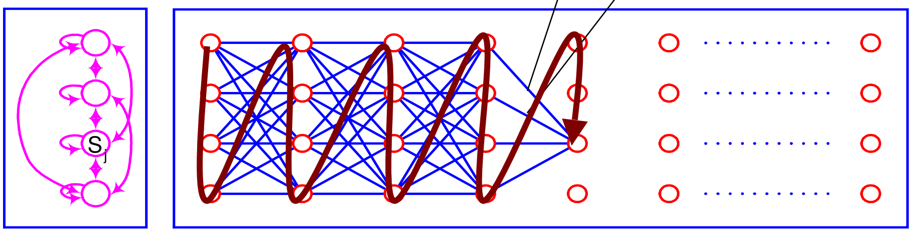
Viterbi Algorithm

- Purpose
  - An analysis for internal processing result
  - The best, the most likely state sequence
  - Internal segmentation

- Viterbi Algorithm
  - Alignment of observation and state transition
  - Dynamic programming technique

Viterbi Path Idea

- Key idea
  - Span a lattice of N states and T times
  - Keep the probability and the previous node of the most probable path coming to each state i at time t
- Recursive path selection
  - Path probability: $\delta_{t+1}(j) = \max_{1 \le i \le N} \delta_t(i) a_{ij} b_j(\mathbf{x}_{t+1})$
  - Path node: $\psi_{t+1}(j) = \arg\max_{1 \le i \le N} \delta_t(i) a_{ij}$

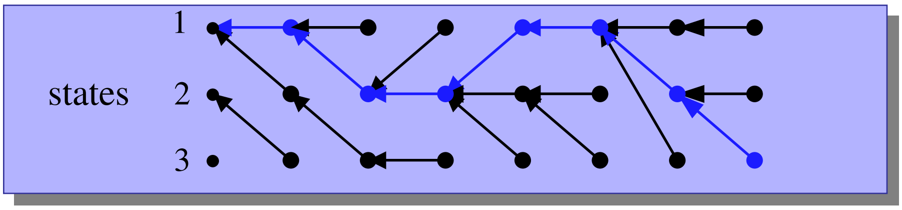


$\delta_t(1) a_{1j} b_j(\mathbf{x}_{t+1})$

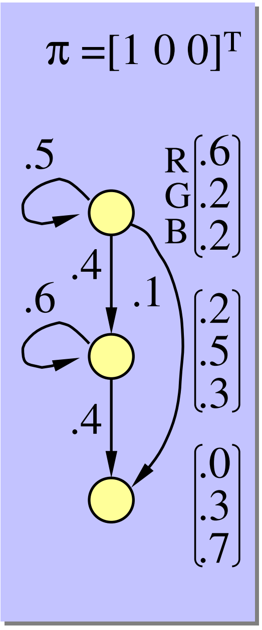
$\delta_t(i) a_{ij} b_j(\mathbf{x}_{t+1})$

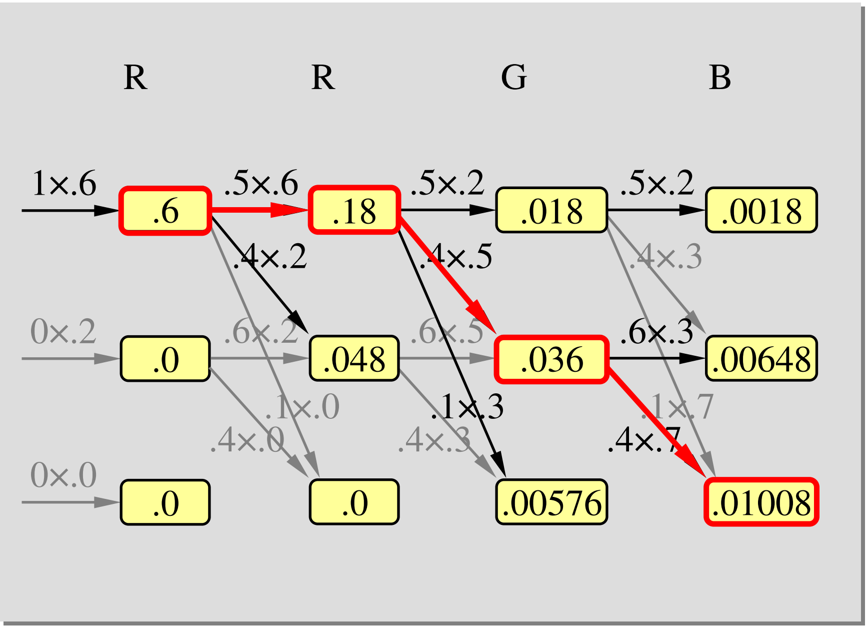
Viterbi Algorithm

- Introduction: $\delta_1(i) = \pi_i b_i(\mathbf{x}_1)$
  $\psi_1(i) = 0$
- Recursion: $\delta_{t+1}(j) = \max_{1 \le i \le N} \delta_t(i) a_{ij} b_j(\mathbf{x}_{t+1})$
  $\psi_{t+1}(j) = \arg\max_{1 \le i \le N} \delta_t(i) a_{ij}$
- Termination: $P^* = \max_{1 \le i \le N} \delta_T(i)$
  $q_T^* = \arg\max_{1 \le i \le N} \delta_T(i)$
- Path backtracking: $q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, \ldots, 1$



Numerical Example: P(RRGB,Q*|λ) [신봉기 03]





Parameter Reestimation

HMM training algorithm
Maximum likelihood estimation
Baum-Welch reestimation

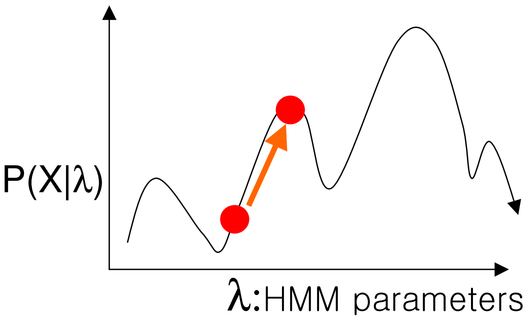
HMM Training Algorithm

- Given an observation sequence $X = \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T$
- Find the model parameter $\lambda^* = (A, B, \pi)$
- s.t. $P(X \mid \lambda^*) \geq P(X \mid \lambda)$ for $\forall \lambda$
  - Adapt HMM parameters maximally to training samples
  - Likelihood of a sample

$$P(X \mid \lambda) = \sum_{Q} P(X \mid Q, \lambda) \boxed{P(Q \mid \lambda)}$$

State transition is hidden!

- NO analytical solution
- *Baum-Welch* reestimation (EM)
  - iterative procedures that locally maximizes P(X|λ)
  - convergence proven
  - MLE statistic estimation

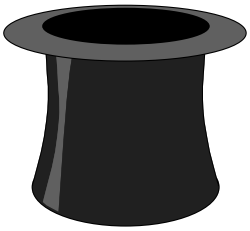


Maximum Likelihood Estimation

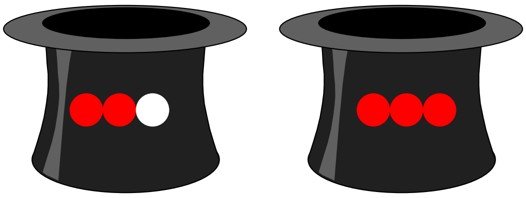
- MLE “selects those parameters that maximizes the probability function of the observed sample.”

- [Definition] Maximum Likelihood Estimate
  - Θ: a set of distribution parameters
  - Given X, Θ* is maximum likelihood estimate of Θ if
  - $f(X|\Theta^*) = \max_\Theta f(X|\Theta)$

MLE Example



- Scenario
  - Known: 3 balls inside urn
  - (some red; some white)
  - Unknown: R = # red balls
  - Observation: ●● (two reds)



- Two models
  - –P (●● |R=2) = $\binom{2}{2}\binom{1}{0}\Big/\binom{3}{2} = \frac{1}{3}$
  - –P (●● |R=3) = $\binom{3}{2}\Big/\binom{3}{2} = 1$

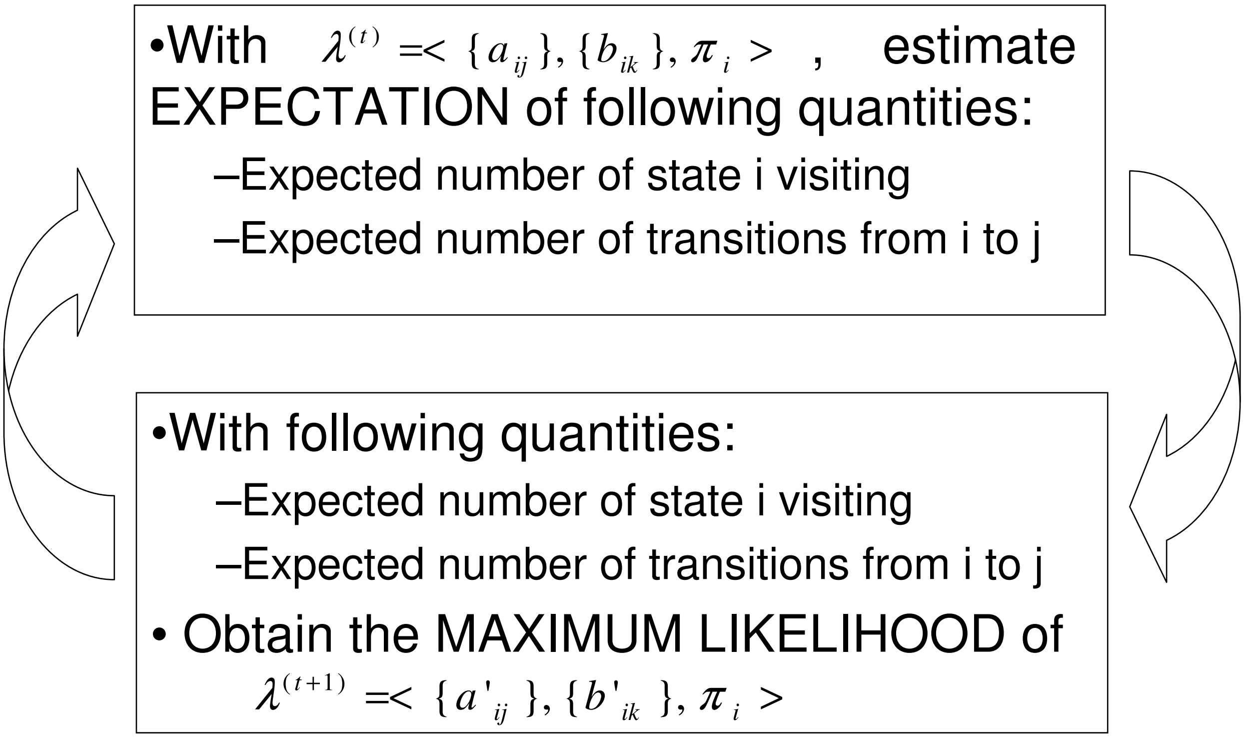
- Which model?
  - $L(\lambda_{R=3}) > L(\lambda_{R=2})$
  - Model(R=3) is our choice

MLE Example (Cont.)

- Model(R=3) is a more likely strategy,
  unless we have a priori knowledge of the system.
- However, without an observation of two red balls
  - No reason to prefer P($\lambda_{R=3}$) to P($\lambda_{R=2}$)

- ML method chooses the set of parameters that maximizes the likelihood of the given observation.
- It makes parameters maximally adapted to training data.

EM Algorithm for Training



•With $\lambda^{(t)} =< \{a_{ij}\}, \{b_{ik}\}, \pi_i >$, estimate EXPECTATION of following quantities:

–Expected number of state i visiting

–Expected number of transitions from i to j

•With following quantities:

–Expected number of state i visiting

–Expected number of transitions from i to j

• Obtain the MAXIMUM LIKELIHOOD of

$\lambda^{(t+1)} =< \{a'_{ij}\}, \{b'_{ik}\}, \pi_i >$

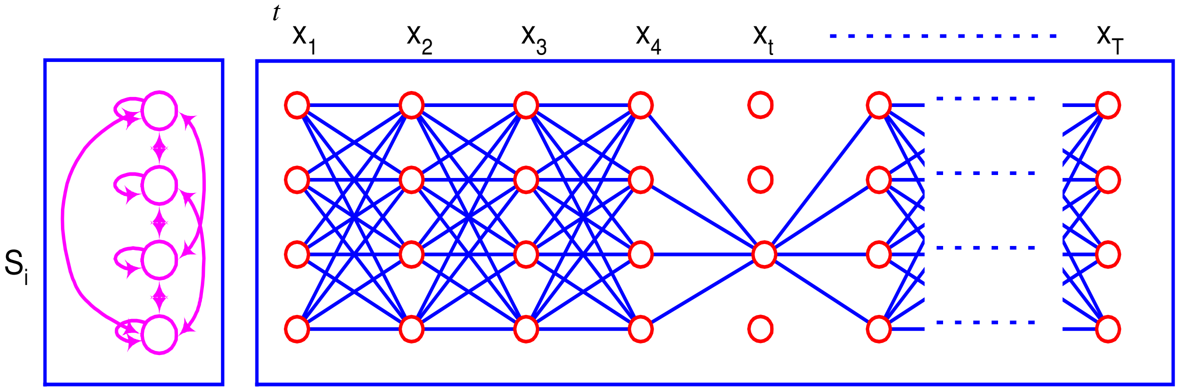
Expected Number of S_i Visiting

$$\gamma_t(i) = P(q_t = S_i \mid X, \lambda)$$

$$= P(q_t = S_i, X \mid \lambda) / P(X \mid \lambda)$$

$$= \frac{\alpha_t(i)\beta_t(i)}{\sum_j \alpha_t(j)\beta_t(j)}$$

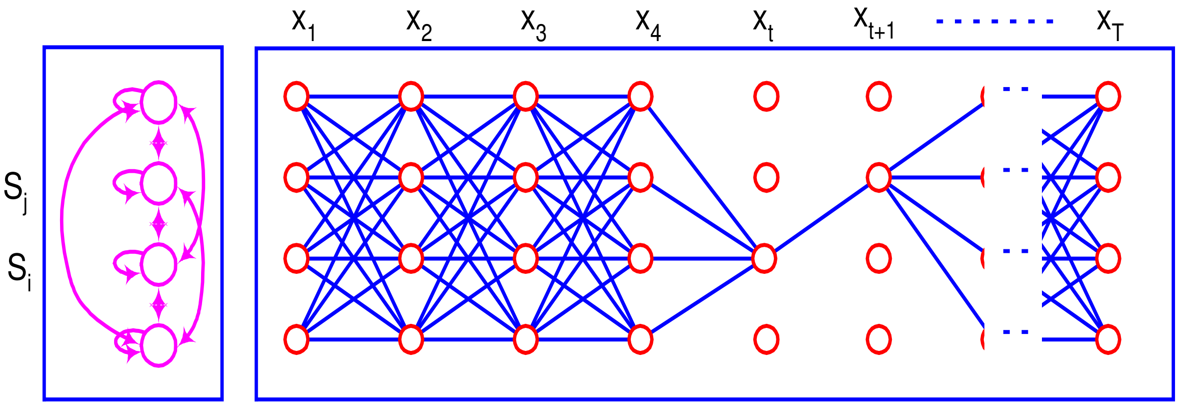
$$\Gamma(i) = \sum_t \gamma_t(i)$$



Expected Number of Transition

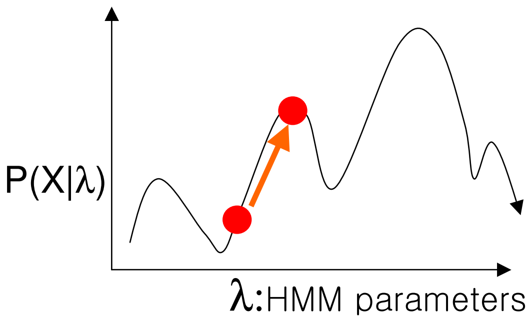
$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j \mid X, \lambda) = \frac{\alpha_t(i)a_{ij}b_j(x_{t+1})\beta_{t+1}(j)}{\sum_i \sum_j \alpha_i(i)a_{ij}b_j(x_{t+1})\beta_{t+1}(j)}$$

$$\Xi(i,j) = \sum_t \xi_t(i,j)$$



Parameter Reestimation

- MLE parameter estimation



$$\overline{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \Gamma_t(i)}$$

$$\overline{b}_i(v_k) = \frac{\sum_{t=1}^{T} \Gamma_t(i)\delta(x_t, v_k)}{\sum_{t=1}^{T} \Gamma_t(i)}$$

$$\overline{\pi}_i = \gamma_1(i)$$

  - Iterative: $P(X \mid \lambda^{(t+1)}) \geq P(X \mid \lambda^{(t)})$
  - convergence proven:
  - arriving local optima

Pattern Classification by HMM

- Pattern classification
- Extension of HMM structure
- Extension of HMM training method
- Practical issues of HMM
- HMM history

Pattern Classification by HMM

- Construct one HMM per each class k
  - $\lambda_1, \cdots, \lambda_N$
- Train each HMM λ_k with samples D_k
  - Baum-Welch reestimation algorithm
- Calculate model likelihood of $\lambda_1, \cdots, \lambda_N$ with observation X
  - Forward algorithm: $P(X \mid \lambda_k)$
- Find the model with maximum *a posteriori* probability

$$\begin{aligned}\lambda^* &= \mathrm{argmax}_{\lambda\mathrm{k}} \ P(\lambda_k \mid X) \\ &= \mathrm{argmax}_{\lambda\mathrm{k}} P(\lambda_k) P(X \mid \lambda_k) / P(X) \\ &= \mathrm{argmax}_{\lambda\mathrm{k}} P(\lambda_k) P(X \mid \lambda_k)\end{aligned}$$

Extension of HMM Structure

- Extension of state transition parameters
  - Duration modeling HMM
    - More accurate temporal behavior
  - Transition-output HMM
    - HMM output functions are attached to transitions rather than states
- Extension of observation parameter
  - Segmental HMM
    - More accurate modeling of trajectories at each state, but more computational cost
  - Continuous density HMM (CHMM)
    - Output distribution is modeled with mixture of Gaussian
  - Semi-continuous HMM
    - Mix of continuous HMM and discrete HMM by sharing Gaussian components

Extension of HMM Training Method

- Maximum Likelihood Estimation (MLE)*
  - maximize the probability of the observed samples

- Maximum Mutual Information (MMI) Method
  - information-theoretic measure
  - maximize average mutual information:

$$I^{*} = \max_{\lambda} \left\{ \sum_{v=1}^{V} \left[\log P(X^{v} \mid \lambda_{v}) - \log \sum_{w=1}^{V} P(X^{w} \mid \lambda_{w}) \right] \right\}$$

  - maximize discrimination power by training models together

- Minimal Discriminant Information (MDI) Method
  - minimize the DI or the cross entropy between pd(signal) and pd(HMM)'s
  - use generalized Baum algorithm

Practical Issues of HMM

- Architectural and behavioral choices
  - the unit of modeling -- design choice
  - type of models: ergodic, left-right, etc.
  - number of states
  - observation symbols;discrete, continuous; mixture number
- Initial estimates
  - A, π : adequate with random or uniform initial values
  - B : good initial estimates are essential for CHMM

Practical Issues of HMM (Cont.)

- Scaling

$$\alpha_t(i) = \prod_{s=1}^{t-1} a_{s,s+1} \prod_{s=1}^{t} b_s(x_s)$$

  - heads exponentially to zero: --> scale by 1 / Si=1,…,N at(i)
- Multiple observation sequences
  - accumulate the expected freq. with weight P(X(k)|l)
- Insufficient training data
  - deleted interpolation with desired model & small model
  - output prob. smoothing (by local perturbation of symbols)
  - output probability tying between different states

HMM History [Roweis]

- Markov('13) and Shannon ('48, '51) studied Markov chains
- Baum et. Al (BP'66, BE'67 …) developed many theories of "probabilistic functions of Markov chains"
- Viterbi ('67) developed an efficient optimal state search algorithm
- Application to speech recognition started
  - Baker('75) at CMU
  - Jelinek's group ('75) at IBM
- Dempster, Laird & Rubin ('77) recognized a general form of the Baum-Welch algorithm

Application of HMM to on-line handwriting recognition with HMM SW

- SW tools for HMM
- Introduction to on-line handwriting recognition
- Data preparation
- Training & testing

SW Tools for HMM

- **HMM toolbox for Matlab**
  - Developed by Kevin Murphy
  - Freely downloadable SW written in Matlab (Hmm… Matlab is not free!)
  - Easy-to-use: flexible data structure and fast prototyping by Matlab
  - Somewhat slow performance due to Matlab
  - Download: http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html

- HTK (Hidden Markov toolkit)
  - Developed by Speech Vision and Robotics Group of Cambridge University
  - Freely downloadable SW written in C
  - Useful for speech recognition research: comprehensive set of programs for training, recognizing and analyzing speech signals
  - Powerful and comprehensive, but somewhat complicate and heavy package
  - Download: http://htk.eng.cam.ac.uk/

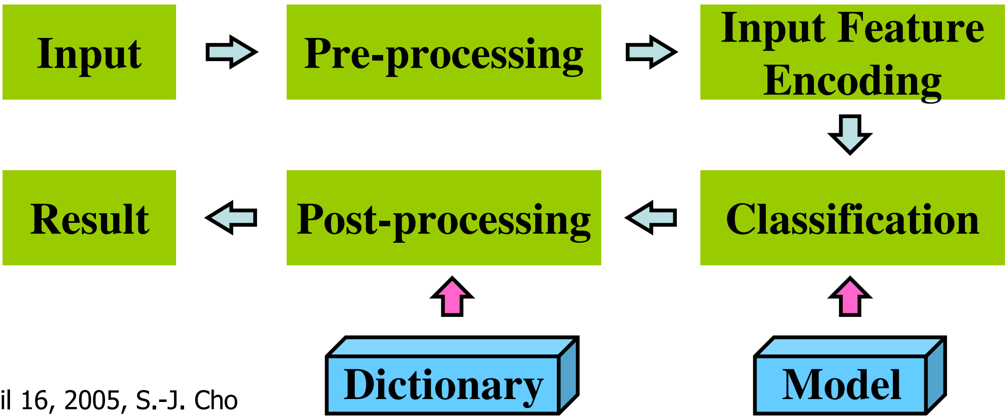
SW Tools: HMM Toolbox for Matlab

- Support training and decoding for
  - Discrete HMMs
  - Continuous HMMs with full, diagonal, or spherical covariance matrix
- 3 Algorithms for discrete HMM
  - Model evaluation (Forward algorithm)
    - Log_likelihood = dhmm_logprob(data, initial state probability, transition probability matrix, observation probability matrix)
  - Viterbi decoding algorithm
    - 1) B = multinomial_prob(data, observation matrix);
      ← B(i,t) = P(y_t | Q_t=i) for all t,i:
    - 2) [path, log_likelihood] = viterbi_path(initial state probability, transition matrix, B)
  - Baum-Welch algorithm
    - [LL, prior2, transmat2, obsmat2] = dhmm_em(data, prior1, transmat1, obsmat1, 'max_iter', 5);

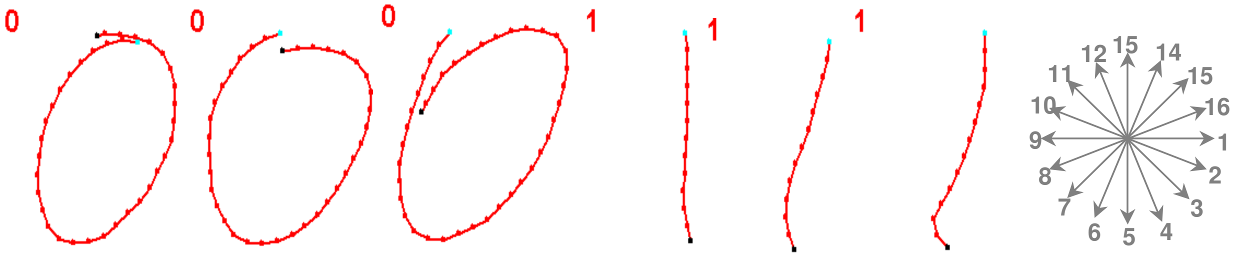
On-line Handwriting Recognition [Sin]

- Handwriting
  - Natural input method to human
  - Sequence of some writing units
  - Temporally ordered
    - Time series of (X,Y) ink points on tablet

- Recognition flow



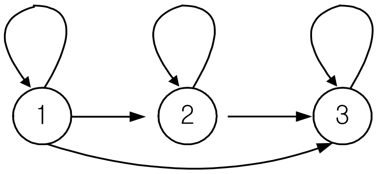
Data Preparation



- Chaincode data set for class ‘0’
  - data0{1} = [9 8 8 7 7 7 6 6 6 5 5 5 4 4 3 2 1 16 15 15 15 15 14 14 14 13 13 13 12 12 11 10 9 9 8]
  - data0{2} = [8 8 7 7 7 6 6 5 5 5 5 4 4 3 2 1 1 16 15 15 15 15 15 14 14 14 14 13 12 11 10 10 9 9 9]
  - data0{3} = [7 6 6 6 6 6 6 5 5 5 4 3 2 1 16 16 16 15 15 15 15 14 14 14 14 14 14 13 11 10 9 9 8 8 8 8 7 7 6 6]
- Chaincode data set for class ‘1’
  - data1{1} = [5 5 5 5 5 5 5 5 5 5 5 4]
  - data1{2} = [5 6 6 6 6 6 6 6 6 5 5 4]
  - data1{3} = [5 5 5 6 6 6 6 6 6 7 6 4 3]

HMM Initialization

- HMM for class ‘0’ and randomly initialization
  - hmm0.prior = [1 0 0];
  - hmm0.transmat = rand(3,3); % 3 by 3 transition matrix
  - hmm0.transmat(2,1) =0; hmm0.transmat(3,1) = 0; hmm0.transmat(3,2) = 0;
  - hmm0.transmat = mk_stochastic(hmm0.transmat);
  - hmm0.transmat



0.20 0.47 0.33

0 0.45 0.55

0 0.00 1.00

  - hmm0.obsmat = rand(3, 16); % # of states * # of observation
  - hmm0.obsmat = mk_stochastic(hmm0.obsmat)

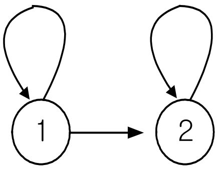
0.02 0.04 0.05 0.00 0.12 0.11 0.13 0.00 0.06 0.09 0.02 0.11 0.06 0.05 0.04 0.08

0.12 0.04 0.07 0.06 0.03 0.03 0.08 0.02 0.11 0.04 0.02 0.06 0.06 0.11 0.01 0.12

0.05 0.04 0.01 0.11 0.02 0.08 0.11 0.10 0.09 0.02 0.05 0.10 0.06 0.00 0.09 0.07

HMM Initialization (Cont.)

- HMM for class ‘1’ and randomly initialization
  - hmm1.prior = [1 0];
  - hmm1.transmat = rand(2,2); % 2 by 2 transition matrix
  - hmm1.transmat(2,1) =0;
  - hmm1.transmat = mk_stochastic(hmm1.transmat);
  - hmm1.transmat



```
 0.03 0.97
   0   1.00
```

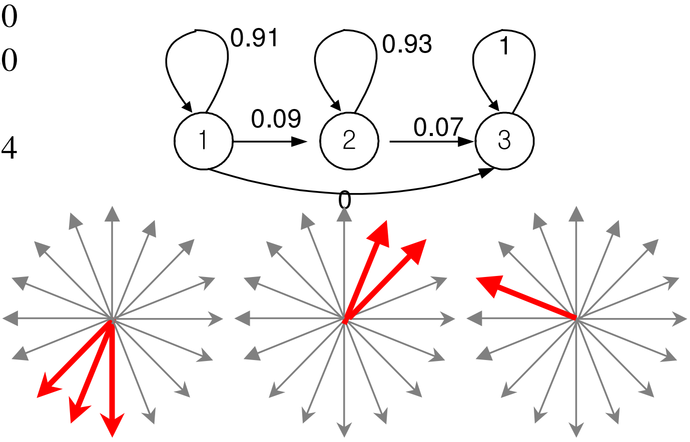
  - hmm1.obsmat = rand(2, 16); % # of states * # of observation
  - hmm1.obsmat = mk_stochastic(hmm1.obsmat)

```
0.05 0.10 0.01 0.06 0.02 0.09 0.06 0.02 0.10 0.04 0.12 0.11 0.03 0.01 0.09 0.11
0.08 0.09 0.06 0.05 0.09 0.10 0.07 0.06 0.12 0.03 0.03 0.12 0.03 0.01 0.03 0.02
```

HMM Training

- Training of model 0
  - [LL0, hmm0.prior, hmm0.transmat, hmm0.obsmat] = dhmm_em(data0, hmm0.prior, hmm0.transmat, hmm0.obsmat)

```
iteration 1, loglik = -365.390770
iteration 2, loglik = -251.112160
…
iteration 9, loglik = -210.991114
```



- Trained result
  - hmm0.transmat

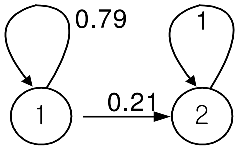
```
0.91 0.09 0.00
0.00 0.93 0.07
0.00 0.00 1.00
```

  - hmm0.obsmat

```
0.00 0.00 0.00 0.00 0.30 0.33 0.21 0.12 0.03 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.09 0.07 0.07 0.11 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.28 0.28 0.11
0.00 0.00 0.00 0.00 0.00 0.06 0.06 0.16 0.23 0.13 0.10 0.10 0.16 0.00 0.00 0.00
```

HMM Training (Cont.)

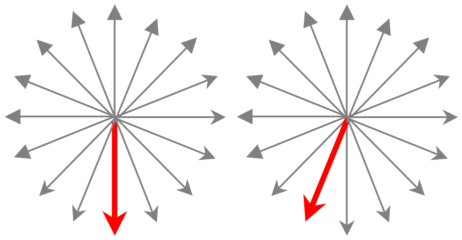
- Training of model 1
  - [LL1, hmm1.prior, hmm1.transmat, hmm1.obsmat] = dhmm_em(data1, hmm1.prior, hmm1.transmat, hmm1.obsmat)
    - iteration 1, loglik = -95.022843
    - …
    - iteration 10, loglik = -30.742533



- Trained model
  - hmm1.transmat

```
0.79 0.21
0.00 1.00
```

  - hmm1.obsmat

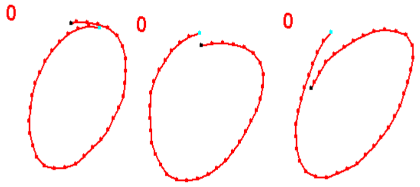


```
0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.04 0.13 0.12 0.66 0.04 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

HMM Evaluation

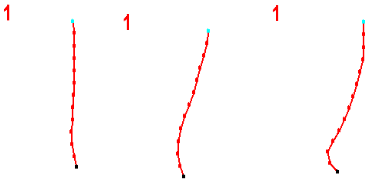
- Evaluation of data 0
  - for dt =1:length(data0)
  - loglike0 = dhmm_logprob(data0{dt}, hmm0.prior, hmm0.transmat, hmm0.obsmat);
  - loglike1 = dhmm_logprob(data0{dt}, hmm1.prior, hmm1.transmat, hmm1.obsmat);
  - disp(sprintf('[class 0: %d-th data] model 0: %.3f, model 1: %.3f',dt, loglike0, loglike1));
  - end

```
[class 0: 1-th data] model 0: -68.969,  model 1: -289.652
[class 0: 2-th data] model 0: -66.370,  model 1: -291.671
[class 0: 3-th data] model 0: -75.649,  model 1: -310.484
```



- Evaluation of data 1

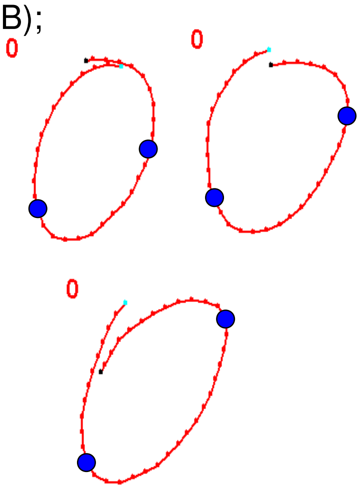
```
[class 0: 1-th data] model 0: -18.676,  model 1: -5.775
[class 0: 2-th data] model 0: -17.914,  model 1: -11.162
[class 0: 3-th data] model 0: -21.193,  model 1: -13.037
```



HMM Decoding

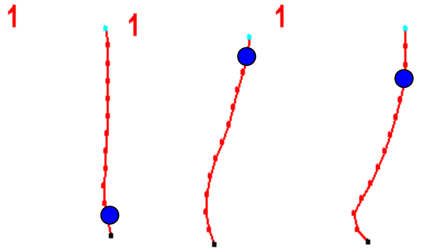
- For data ‘0’, get the most probable path
  - for dt =1:length(data0)
  - B = multinomial_prob(data0{dt}, hmm0.obsmat);
  - path = viterbi_path(hmm0.prior, hmm0.transmat, B);
  - disp(sprintf('%d', path));
  - end

```
11111111111122222222222223333333333
11111111111122222222222222233333333
11111111111222222222222222223333333333333
```



- For data ‘1’, get the most probable path

```
111111111112
122222222222
1112222222222
```



Summary

- Markov model
  - 1-st order Markov assumption on state transition
  - ‘Visible’: observation sequence determines state transition seq.
- Hidden Markov model
  - 1-st order Markov assumption on state transition
  - ‘Hidden’: observation sequence may result from many possible state transition sequences
  - Fit very well to the modeling of spatial-temporally variable signal
  - Three algorithms: model evaluation, the most probable path decoding, model training
- Example of HMM application to on-line handwriting recognition
  - Use HMM tool box for Matlab

References

- Hidden Markov Model
  - L.R. Rabiner, “A Tutorial to Hidden Markov Models and Selected Applications in Speech Recognition”, *IEEE Proc*. pp. 267-295, 1989
  - L.R. Bahl et. al, “A Maximum Likelihood Approach to Continuous Speech Recognition”, *IEEE PAMI*, pp. 179-190, May. 1983
  - M. Ostendorf, “From HMM’s to Segment Models: a Unified View of Stochastic Modeling for Speech Recognition”, *IEEE SPA*, pp 360-378, Sep., 1996
- HMM Tutorials
  - 신봉기, “HMM Theory and Applications “, *2003컴퓨터비젼및패턴 인식연구회 춘계워크샵 튜토리얼*
  - Sam Roweis, “Hidden Markov Models (**SCIA Tutorial 2003**)”, *http://www.cs.toronto.edu/~roweis/notes/scia03h.pdf*
  - Andrew Moore, “Hidden Markov Models”, *http://www-2.cs.cmu.edu/~awm/tutorials/hmm.html*

References (Cont.)

- HMM SW
  - Kevin Murphy, “HMM toolbox for Matlab”, freely downloadable SW written in Matlab, http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html
  - Speech Vision and Robotics Group of Cambridge University, “HTK (Hidden Markov toolkit)”, freely downloadable SW written in C, http://htk.eng.cam.ac.uk/
- Online Character Recognition
  - C.C. Tappert et. al, “The state of the Art in On-Line Handwriting Recognition”, *IEEE PAMI*, pp. 787-808, Aug, 1990
  - B.K. Sin and J.H. Kim, “Ligature Modeling for Online Cursive Script Recognition”, *IEEE PAMI*, pp. 623-633,Jun, 1997
  - S.-J. Cho and J.H. Kim, “Bayesian Network Modeling of Character Components and Their Relationships for On-line Handwriting Recognition”, *Pattern Recognition*, pp253-264, Feb. 2004
  - J. Hu, et. al, “Writer Independent On-line Handwriting Recognition Using an HMM Approach”, *Pattern Recognition*, pp 133-147, 2000

Appendix: Matlab Code (I)

```
% chaincode data set for class '0'
data0{1} = [9 8 8 7 7 7 6 6 6 5 5 5 4 4 3 2 1 16 15 15 15 15 14 14 14 13 13 13 12 12 11
    10 9 9 8 ];
data0{2} = [8 8 7 7 7 6 6 5 5 5 5 4 4 3 2 1 1 16 15 15 15 15 15 14 14 14 14 13 12 11 10
    10 9 9 9 ];
data0{3} = [7 6 6 6 6 6 6 5 5 5 4 3 2 1 16 16 16 15 15 15 15 14 14 14 14 14 14 13 11 10
    9 9 8 8 8 8 7 7 6 6 ];
% chaincode data set for class ‘1’
data1{1} = [5 5 5 5 5 5 5 5 5 5 5 4 ];
data1{2} = [5 6 6 6 6 6 6 6 6 5 5 4 ];
data1{3} = [5 5 5 6 6 6 6 6 6 7 6 4 3];

% HMM for class '0' and random initialization of parameters
hmm0.prior = [1 0 0];
hmm0.transmat = rand(3,3); % 3 by 3 transition matrix
hmm0.transmat(2,1) =0; hmm0.transmat(3,1) = 0; hmm0.transmat(3,2) = 0;
hmm0.transmat = mk_stochastic(hmm0.transmat);
hmm0.transmat
hmm0.obsmat = rand(3, 16); % # of states * # of observation
hmm0.obsmat = mk_stochastic(hmm0.obsmat)
```

Appendix: Matlab Code (2)

```
% HMM for class '1' and random initialiation of parameters
hmm1.prior = [1 0 ];
hmm1.transmat = rand(2,2); % 2 by 2 transition matrix
hmm1.transmat(2,1) =0;
hmm1.transmat = mk_stochastic(hmm1.transmat);
hmm1.transmat

hmm1.obsmat = rand(2, 16); % # of states * # of observation
hmm1.obsmat = mk_stochastic(hmm1.obsmat)

% Training of HMM model 0 (Baum-Welch algorithm)
[LL0, hmm0.prior, hmm0.transmat, hmm0.obsmat] = dhmm_em(data0, hmm0.prior,
    hmm0.transmat, hmm0.obsmat)
% smoothing of HMM observation parameter: set floor value 1.0e-5
hmm0.obsmat = max(hmm0.obsmat, 1.0e-5);

% Training of HMM model 1 (Baum-Welch algorithm)
[LL1, hmm1.prior, hmm1.transmat, hmm1.obsmat] = dhmm_em(data1, hmm1.prior,
    hmm1.transmat, hmm1.obsmat)
% smoothing of HMM observation parameter: set floor value 1.0e-5
hmm1.obsmat = max(hmm1.obsmat, 1.0e-5);
```

Appendix: Matlab Code(3)

```
% Compare model likelihood
%Evaluation of class '0' data
for dt =1:length(data0)
   loglike0 = dhmm_logprob(data0{dt}, hmm0.prior, hmm0.transmat, hmm0.obsmat);
   loglike1 = dhmm_logprob(data0{dt}, hmm1.prior, hmm1.transmat, hmm1.obsmat);
   disp(sprintf('[class 0: %d-th data] model 0: %.3f,  model 1: %.3f',dt,  loglike0, loglike1));
end

for dt =1:length(data1)
   loglike0 = dhmm_logprob(data1{dt}, hmm0.prior, hmm0.transmat, hmm0.obsmat);
   loglike1 = dhmm_logprob(data1{dt}, hmm1.prior, hmm1.transmat, hmm1.obsmat);
   disp(sprintf('[class 1: %d-th data] model 0: %.3f,  model 1: %.3f',dt,  loglike0, loglike1));
end
```

Appendix: Matlab Code (4)

```
%Viterbi path decoding
%First you need to evaluate B(i,t) = P(y_t | Q_t=i) for all t,i:
path0 = cell(1, length(data0));
for dt =1:length(data0)
   B = multinomial_prob(data0{dt}, hmm0.obsmat);
   path0{dt} = viterbi_path(hmm0.prior, hmm0.transmat, B);
   disp(sprintf('%d', path0{dt}));
end

path1 = cell(1, length(data1));
for dt =1:length(data1)
   B = multinomial_prob(data1{dt}, hmm1.obsmat);
   path1{dt} = viterbi_path(hmm1.prior, hmm1.transmat, B);
   disp(sprintf('%d', path1{dt}));
end
```